COPERTINA

INDICE

1. Abstract
2. Linguaggio dei Segni
3. MediaPipe
4. Obiettivo (Risultato Atteso)
5. Implementazione
   1. StoreGesture.py
   2. Recogniser.py
6. Risultato Ottenuto
7. Condizioni di Luminosità
8. Requisiti Hardware and Software
9. Abstract

TODO

1. Linguaggio dei Segni

Le lingue dei segni sono lingue che veicolano i propri significati attraverso un sistema codificato di segni delle mani, espressioni del viso e movimenti del corpo. Sono utilizzate dalle comunità dei segnanti, a cui appartengono, in maggioranza, persone sorde.

La comunicazione attraverso la lingua dei segni avviene tramite il canale visivo-gestuale, producendo dei segni precisi, compiuti con una o entrambe le mani, aventi un significato specifico e assodato, come avviene per le parole.

I segni di ogni lingua dei segni possono essere scomposti in 4 componenti essenziali:

* movimento,
* orientamento,
* configurazione,
* luogo

(ossia le quattro componenti manuali del segno)

Le lingue dei segni sono lingue complete con una propria grammatica e un proprio dizionario, ma ne parliamo al plurale in quanto, ad ogni nazione, corrispondono diverse lingue dei segni; in Italia troviamo la **Lingua dei Segni Italiana** (**LIS**), negli Stati Uniti la **Lingua dei Segni Americana** (**ASL**), nel [Regno Unito](https://it.wikipedia.org/wiki/Regno_Unito) la [**Lingua dei Segni Britannica**](https://it.wikipedia.org/wiki/Lingua_dei_segni_britannica) (**BSL**), … e così via.

Motivo per cui è stato deciso di codificare una lista di segni "internazionali", che facilitasse il superamento delle [barriere](https://it.wikipedia.org/wiki/Barriera_linguistica) linguistiche.

Prende il nome di Signuno o Gestuno ed è la **Lingua dei Segni Internazionale**, sviluppata dalla World Federation of Deaf negli anni ’50, per permette la comunicazione tra persone sorde anche se di diversa nazionalità, e quindi usanti diverse lingue dei segni.

Trattandosi di una proposta esclusivamente lessicale, fu utilizzato, talvolta, in contesti di incontri internazionali, senza però acquisire mai le caratteristiche di una vera e propria lingua.

In Fig. 2.1. è possibile osservare i segni rappresentanti l’alfabeto del Signuno.



Fig. 2.1

1. MediaPipe

MediaPipe è una libreria multipiattaforma sviluppata da Google che fornisce incredibili soluzioni di Machine Learning pronte all'uso per attività di visione artificiale. Più specificatamente è un framework basato su grafi per la creazione di pipeline di machine learning multimodali (video, audio e sensori).

Tramite l’uso di Mediapipe e OpenCV è stato possibile implementare un prototipo scalabile di un Riconoscitore di gesti che abbiamo poi adattato al riconoscimento delle lettere dell’alfabeto della lingua dei segni internazionale.

* 1. MediaPipe Hands

Nonostante, per una persona, il riconoscimento di una mano e i suoi movimenti sia un task piuttosto semplice, ciò non è altrettanto corretto dal punto di vista di una macchina, infatti la percezione della mano in tempo reale è un compito di Computer Vision decisamente impegnativo, poiché le mani spesso si occludono a vicenda (ad es. occlusioni di dita/palmo, tremori e orientamento della mano) e ci si ritrova spesso in mancanza di pattern ad alto contrasto.

A tal proposito **MediaPipe Hands** è una soluzione di tracciamento di mani e dita ad alta fedeltà. Utilizza il Machine Learning (ML) per dedurre 21 punti di riferimento 3D di una mano da un solo fotogramma.

* + 1. Pipeline di Machine Learning

MediaPipe Hands utilizza una pipeline di Machine Learning composta da più modelli che lavorano insieme:

* Un **Palm Detection Model** che opera sull'immagine completa e restituisce un riquadro di delimitazione della mano orientato.
* Un **Hand Landmark Model** che opera sulla regione dell'immagine ritagliata definita dal rilevatore del palmo e restituisce punti chiave della mano 3D ad alta fedeltà.

Fornire l'immagine della mano accuratamente ritagliata all’**Hand Landmark Model** riduce drasticamente la necessità di aumentare i dati (ad esempio rotazioni, traslazione e scala) e consente invece alla rete di dedicare la maggior parte della sua capacità all'accuratezza della previsione delle coordinate.

* + 1. Palm Detection Model

La mancanza di motivi ad alto contrasto nelle mani rende relativamente difficile rilevarle in modo affidabile solo dalle loro caratteristiche visive. Invece, fornire un contesto aggiuntivo, come le caratteristiche del braccio, del corpo o della persona, aiuta la localizzazione accurata della mano.

Il **Palm Detection Model** nasce dal fatto che stimare i riquadri di delimitazione di oggetti rigidi come palmi e pugni è significativamente più semplice rispetto al rilevamento di mani con dita articolate. Inoltre, i palmi possono essere modellati utilizzando riquadri di delimitazione quadrati (“*ancore*” nella terminologia ML) ignorando altre proporzioni e quindi riducendo il numero di ancoraggi di un fattore 3-5. In secondo luogo, un estrattore di funzionalità codificatore-decodificatore viene utilizzato per una maggiore consapevolezza del contesto della scena anche per piccoli oggetti. Infine, la perdita focale viene ridotta al minimo durante l'allenamento per supportare una grande quantità di ancoraggi risultanti dalla varianza su larga scala.

Con le tecniche di cui sopra, si ottiene una precisione media del 95,7% nel rilevamento del palmo.

* + 1. Hand Landmark Model

Dopo il rilevamento del palmo sull'intera immagine, il successivo **Hand Landmark Model** esegue una precisa localizzazione dei punti chiave di 21 coordinate 3D delle nocche della mano all'interno delle regioni della mano rilevate tramite regressione, ovvero la previsione diretta delle coordinate. Il modello apprende una rappresentazione coerente della posa della mano interna ed è robusto anche per mani parzialmente visibili e in presenza di auto-occlusioni.

Per ottenere il risultato sono state annotate manualmente ~30.000 immagini del mondo reale con 21 coordinate 3D, come mostrato in Fig.1.1 (il valore Z viene preso dalla mappa di profondità dell'immagine, se esiste, per ogni coordinata corrispondente). Per coprire meglio le possibili pose della mano e fornire un'ulteriore supervisione sulla natura della geometria della mano, viene eseguito anche il rendering di un modello di mano sintetico di alta qualità su vari sfondi e lo mappiamo alle corrispondenti coordinate 3D.

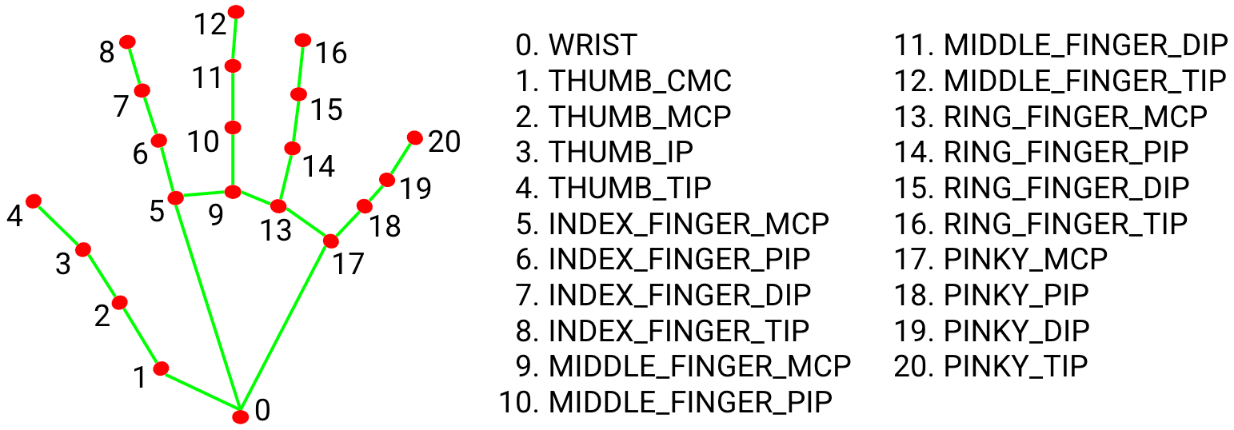


Fig. 3.1

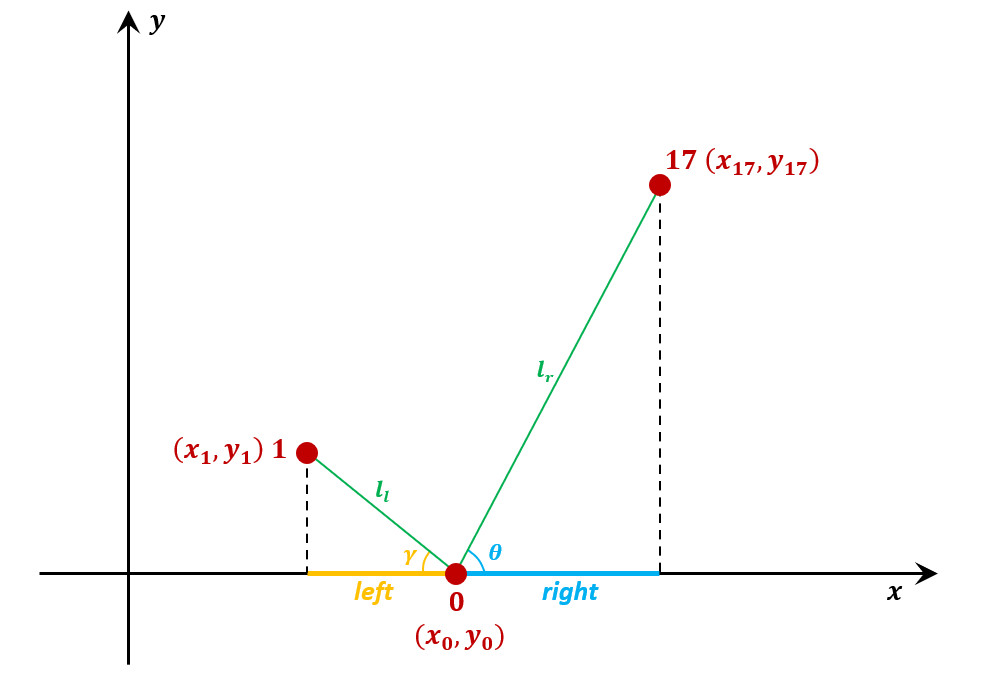
* + 1. Configuration Options
* **STATIC\_IMAGE\_MODE**: Se impostato su *false*, la soluzione tratta le immagini di input come un flusso video. Tenterà di rilevare le mani nelle prime immagini di input e, in caso di rilevamento riuscito, localizzerà ulteriormente i punti di riferimento della mano. Nelle immagini successive, una volta rilevate tutte le mani e localizzati i corrispondenti punti di riferimento delle mani, tiene semplicemente traccia di tali punti di riferimento senza invocare un altro rilevamento fino a quando non perde traccia di tutte le mani. Ciò riduce la latenza ed è ideale per l'elaborazione di fotogrammi video. Se impostato su *true*, il rilevamento delle mani viene eseguito su ogni immagine di input, ideale per l'elaborazione di un batch di immagini statiche, possibilmente non correlate. Il valore di default è *false*. FALSE
* **MAX\_NUM\_HANDS**: Il numero massimo da mani da rilevare. Il valore di default è 2. 1
* **MODEL\_COMPLEXITY**:
  + 1. Output

1. Obiettivo

TODO

1. Implementazione
   1. HandTrackingModule.py
      1. Orientamento

Per semplicità consideriamo il seguente sistema di riferimento, rappresentando i nodi 0, 1 e 17 della mano, l’asse x è coerente col modello utilizzato da MediaPipe, mentre l’asse y no, in quanto la massima coordinata y, in MediaPipe, si raggiunge in corrispondenza del punto 0. Tuttavia, ai fini del calcolo dell’orientamento, questa differenza è ininfluente.



Per il Teorema di Pitagora abbiamo che:

Quindi si ha che:

Sostituendo:

Il coseno al quadrato è una funzione limitata nell’intervallo per cui basta trovare il giusto intervallo entro cui la mano risulta orientata più o meno verticalmente.

L’intervallo di è più ampio in quanto la distanza che intercorre tra i nodi 0 e 1 è minore rispetto a quella fra i nodi 0 e 17.

* 1. Recogniser.py
  2. StoreGestures.py

1. Risultato Ottenuto

TODO

1. Condizioni di Luminosità

TODO

1. Requisiti Hardware e Software

TODO

1. NOTE

# finds the min in error array  
for i in range(0, len(errorArray), 1):  
 if errorArray[i] < errorMin:  
 errorMin = errorArray[i]  
 minIndex = i  
if errorMin < tolMN:  
 if gestNames[minIndex] == 'M' or gestNames[minIndex] == 'N':  
 gesture = gestNames[minIndex]  
 print(gestNames[minIndex])  
 elif errorMin < tol:  
 gesture = gestNames[minIndex]  
 elif errorMin >= tol:  
 gesture = 'Unknown'  
return gesture